**R Reference Card**

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07

Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

### Getting help

Most R functions have online documentation. `help(topic)` documentation on topic `?topic` id. `help.search("topic")` search the help system `apropos("topic")` the names of all objects in the search list matching the regular expression "topic"

### Input and output

`load()` load datasets written with `save`

`data(x)` loads specified data sets

`library(x)` load add-on packages

`read.table(file)` reads a file in table format and creates a data frame from it; the default separator `sep=""` is any whitespace; use `header=TRUE` to read the first line as a header of column names; use `as.is=TRUE` to prevent character vectors from being converted to factors; `use.column.names=TRUE` to use column names from the file if present; `na.strings=NA` to treat a NA as a string.

`read.csv("filename",header=TRUE)` id. but with defaults set for reading comma-delimited files

`read.delim("filename",header=TRUE)` id. but with defaults set for reading tab-delimited files

`read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE)` read a table of fixed width/formatted data into a `data.frame`; `widths` is an integer vector, giving the widths of the fixed-width fields

`save(file,...)` saves the specified objects (...) in the XDR platform-independent binary format

`save.image(file)` saves all objects

`cat(...) file="",sep="")` prints the arguments after coercing to character; `sep` is the character separator between arguments

`print(a,...)` prints its arguments; generic, meaning it can have different methods for different objects

`format(x,...)` format an R object for pretty printing

`write.table(x,file="",row.names=TRUE,col.names=TRUE,sep="")` prints x after converting to a data frame; if `quote=TRUE`, character or factor columns are surrounded by quotes; `*` `sep` is the field separator; `eol` is the end-of-line separator; `na` is the string for missing values; use `col.names=NA` to add a blank column header to get the column headers aligned correctly for spreadsheet input

`sink(file)` output to `file`, until `sink()`

Most of the I/O functions have a file argument. This can often be a character string naming a file or a connection. `file=` means the standard input or output. Connections can include files, pipes, zipped files, and R variables.

On windows, the file connection can also be used with description *"clipboard"*. To read a table copied from Excel, use `x <- read.delim("clipboard")`

To write a table to the clipboard for Excel, use `write.table(x,\"\"clipboard\",sep=\"\",col.names=\"\")`

For database interaction, see packages RODBC, DBI, RMySQL, RPGSQL, and ROracle. See packages XML, hdfs, netCDF for reading other file formats.

### Data creation

`c(...)` generic function to combine arguments with the default forming a vector; with `recursive=TRUE` descends through lists combining all elements into one vector

`from:to` generates a sequence: "" has operator priority; `1:4 + 1` is `2,3,4,5`

`seq(from,to)` generates a sequence `by` specifies increment; `length` specifies desired length

`seq(along=x)` generates a sequence

`rep(x,times)` replicates `x` `times`; use `each` to repeat "each" element of `x` `times`; `rep(c(1,2,3),each=2)` is `1 2 3 1 2 3`

`data.frame(...)` create a data frame of the named or unnamed arguments; `data.frame(v=1:4,c=c("a","b","c","d"),n=10)`; shorter vectors are recycled to the length of the longest

`list(...)` create a list of the named or unnamed arguments; `list(a=c(1,2),b=\"hi\",c=\")`

`array(x,dim=)` array with data `x`; specify dimensions like `dim=c(3,2)`

`dimnames(x)` get or set the dimension names of an object

`row()` and `nrow(x)` number of rows; `NROW(x)` is the same but treats a vector as a one-row matrix

`ncol(x)` and `NCOL(x)` id. for columns

`class(x)` get or set the class of `x`; `class(x) <- \"myclass\"` changes the class attribute of `x`

`attributes(obj)` get or set the list of attributes of `obj`

### Slicing and extracting data

`Indexing vectors

`x[n]` `nth` element

all but the `n`th element

first `n` elements

elements from `n` to `end`

specific elements

element named "name"

all elements greater than `n`

all elements between `n` and `m`

all elements and `n` elements in the given set

`Indexing lists

`x[n]` list with elements `n`

`x[n]` `nth` element of the list

`x[\"name\"]` element of the list named "name"

`x$name` id.

`Indexing matrices

`x[i,j]` element at row `i`, column `j`

`x[i]` row `i`

`x[,j]` column `j`

`x[,c(1,3)]` columns 1 and 3

`x[\"name\"]` row named "name"

Indexing data frames (matrix indexing plus the following)

`x[\"name\"]` column named "name"

`x$name` id.

### Variable conversion

`as.array(x), as.data.frame(x), as.numeric(x), as.logical(x), as.complex(x), as.character(x), ...` convert type; for a complete list, use `methods(as)`

### Variable information

`is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x), ...` test for type; for a complete list, use `methods(is)`

`length(x)` number of elements in `x`

`dim(x)` Retrieve or set the dimension of an object; `dim(x) <- c(3,2)`

`dimnames(x)` Retrieve or set the dimension names of an object

`nrow(x)` number of rows; `NROW(x)` is the same but treats a vector as a one-row matrix

Data selection and manipulation

`which.max(x)` returns the index of the greatest element of `x`

`which.min(x)` returns the index of the smallest element of `x`

`rev(x)` reverses the elements of `x`

`sort(x)` sorts the elements of `x` in increasing order; to sort in decreasing order: `rev(sort(x))`

`cut(x,breaks)` divides `x` into intervals (factors); `breaks` is the number of cut intervals or a vector of cut points

`match(x,y)` returns a vector of the same length than `x` with the elements of `x` which are in `y` (NA otherwise)

which(x=a) returns a vector of the indices of `x` if the comparison operation is true (TRUE), in this example the values of `l` for which `x[1]` is a (the argument of this function must be a variable of mode logical)

choose(n,k) computes the combinations of k elements among n repetitions `n!/(k!(n-k)!)`

`na.omit(x)` suppresses the observations with missing data (NA) (suppresses the corresponding line if `x` is a matrix or a data frame)

`na.fail(x)` returns an error message if `x` contains at least one NA

### Data selection and manipulation

which.max(x)` returns the index of the greatest element of `x`

which.min(x)` returns the index of the smallest element of `x`

rev(x)` reverses the elements of `x`

sort(x)` sorts the elements of `x` in increasing order; to sort in decreasing order: `rev(sort(x))`

`cut(x,breaks)` divides `x` into intervals (factors); `breaks` is the number of cut intervals or a vector of cut points

match(x,y)` returns a vector of the same length than `x` with the elements of `x` which are in `y` (NA otherwise)

which(x=a) returns a vector of the indices of `x` if the comparison operation is true (TRUE), in this example the values of `l` for which `x[1]` is a (the argument of this function must be a variable of mode logical)

choose(n,k) computes the combinations of k elements among n repetitions `n!/(k!(n-k)!)`

na.omit(x)` suppresses the observations with missing data (NA) (suppresses the corresponding line if `x` is a matrix or a data frame)

na.fail(x)` returns an error message if `x` contains at least one NA

### Data selection and manipulation

which.max(x)` returns the index of the greatest element of `x`

which.min(x)` returns the index of the smallest element of `x`

rev(x)` reverses the elements of `x`

sort(x)` sorts the elements of `x` in increasing order; to sort in decreasing order: `rev(sort(x))`

`cut(x,breaks)` divides `x` into intervals (factors); `breaks` is the number of cut intervals or a vector of cut points

match(x,y)` returns a vector of the same length than `x` with the elements of `x` which are in `y` (NA otherwise)

which(x=a) returns a vector of the indices of `x` if the comparison operation is true (TRUE), in this example the values of `l` for which `x[1]` is a (the argument of this function must be a variable of mode logical)

choose(n,k) computes the combinations of k elements among n repetitions `n!/(k!(n-k)!)`

na.omit(x)` suppresses the observations with missing data (NA) (suppresses the corresponding line if `x` is a matrix or a data frame)

na.fail(x)` returns an error message if `x` contains at least one NA
unique(x) if x is a vector or a data frame, returns a similar object but with the duplicate elements suppressed

table(x) returns a table with the numbers of the different values of x (typically for integers or factors)

subset(x, ...) returns a selection of x with respect to criteria (...), typically comparisons: x[V1 < 10] if x is a data frame, the option select gives the variables to be kept or dropped using a minus sign

table(x, size) resample randomly and without replacement size elements in the vector x, the option replace = TRUE allows to resample with replacement

prop.table(x, margin=) table entries as fraction of marginal table

Math

sin, cos, tan, asin, acos, atan, atan2, log, log10, exp

max(x) maximum of the elements of x

min(x) minimum of the elements of x

range(x) id. then c(min(x), max(x))

sum(x) sum of the elements of x

diff(x) lagged and iterated differences of vector x

prod(x) product of the elements of x

mean(x) mean of the elements of x

median(x) median of the elements of x

quantile(x, probs=) sample quantiles corresponding to the given probabilities (defaults to 0.25, 0.5, 0.75, 1)

weighted.mean(x, w) mean of x with weights w

rank(x) ranks of the elements of x

var(x) or cov(x) variance of the elements of x (calculated on n - 1); if x is a matrix or a data frame, the variance-covariance matrix is calculated

sd(x) standard deviation of x

cor(x) correlation matrix of x if it is a matrix or a data frame (1 if x is a vector)

var(x, y) or cov(x, y) covariance between x and y, or between the columns of x and those of y if they are matrices or data frames

cor(x, y) linear correlation between x and y, or correlation matrix if they are matrices or data frames

round(x, n) rounds the elements of x to n decimals

log(x, base) computes the logarithm of x with base base

scale(x) if x is a matrix, centers and reduces the data; to center only use the option center=FALSE, to reduce only scale=FALSE (by default center=TRUE, scale=TRUE)

pmin(x, y, ...) a vector whose element is the minimum of x[i], y[i], ...

pmax(x, y, ...) id. for the maximum

cumsum(x) a vector whose element is the sum from x[1] to x[i]

cumprod(x) id. for the product

cummin(x) id. for the minimum

cummax(x) id. for the maximum

union(x, y) intersect(x, y), setdiff(x, y), setequal(x, y), is.element(el, set) "set" functions

Re(x) real part of a complex number

Im(x) imaginary part

Mod(x) modulus; abs(x) is the same

Arg(x) angle in radians of the complex number

Conj(x) complex conjugate

convolve(x, y) compute the several kinds of convolutions of two sequences

fft(x) Fast Fourier Transform of an array

mvfft(x) FFT of each column of a matrix

filter(x, filter) applies linear filtering to a univariate time series or to each series separately of a multivariate time series

Many math functions have a logical parameter na.rm=FALSE to specify missing data (NA) removal.

Matrices

t(x) transpose

diag(x) diagonal

%*% matrix multiplication

solve(a, b) solves a %*% x = b for x

solve(a) matrix inverse of a

rowsum(x) sum of rows for a matrix-like object; rowSums(x) is a faster version

colsum(x), colSums(x) id. for columns

rowMeans(x) fast version of row means

colMeans(x) id. for columns

Advanced data processing

apply(X, INDEX, FUN=) a vector or array or list of values obtained by applying a function FUN to margins (INDEX) of X

lapply(X, FUN=) apply each element of the list X

tapply(X, INDEX, FUN=) apply each cell of a ragged array given by X with indexes INDEX

by(data, INDEX, FUN) apply FUN to data frame data subsetted by INDEX

merge(a, b) merge two data frames by common columns or row names

xtabs(a, b, data=x) a contingency table from cross-classifying factors

aggregate(x, by, FUN) splits the data frame x into subsets, computes summary statistics for each, and returns the result in a convenient form; by is a list of grouping elements, each as long as the variables in x

stack(x, ...) transform data available as separate columns in a data frame or list into a single column

unstack(x, ...) inverse of stack()

reshape(x, ...) reshapes a data frame between 'wide' format with data frame and 'long' format with repeated measurements in separate columns of the same record and 'long' format with the repeated measurements in separate records; use (direction="wide") or (direction="long")

Strings

paste(...) concatenate vectors after converting to character; sep= is the string to separate terms (a single space is the default); collapse= is an optional string to separate "collapsed" results

substr(x, start, stop) substrings in a character vector; can also as sign, asubstr(x, start, stop) <- value

split(x, split) split x according to the substring split

grep(pattern, x) searches for matches to pattern within x; see ?regex

gsub(pattern, replacement, x) replacement of matches determined by regular expression matching sub () is the same but only replaces the first occurrence.

tolower(x) convert to lowercase

toupper(x) convert to uppercase

match(x, table) a vector of the positions of first matches for the elements of x among table

t x in? table id. but returns a logical vector

pmatch(x, table) partial matches for the elements of x among table

nchar(x) number of characters

Dates and Times

The class Date has dates without times. POSIXxct has dates and times, including time zones. Comparisons (e.g. >), seq(), and difftime() are useful. See also package chron.

as.Date(x) and as.POSIXct(x) convert to the respective class; format(dt) converts to a string representation. The default string format is "2001-02-21". These accept a second argument to specify a format for conversion. Some common formats are:

%a, %A Abbreviated and full weekday name.
%b, %B Abbreviated and full month name.
%d Day of the month (01-31).
%m Day of year (001-366).
%M Month (01-12).
%H Hour (00-23).
%I Hours (01-12).
%j Day of year (001-366).
%u Weekday (1-7, Monday is 1).
%V Year without century (00-99). Don’t use.
%y Year with century.
%z (output only.) Offset from Greenwich: 0000 is 8 hours west of.
%Z (output only.) Time zone as a character string (empty if not available).

Where leading zeros are shown they will be used on output but are optional on input. See ?strftime.

Plotting

plot(x) plot of the values of x (on the y-axis) ordered on the x-axis

plot(x, y) bivariate plot of x (on the x-axis) and y (on the y-axis)

hist(x) histogram of the frequencies of x

barplot(x) histogram of the values of x; use horiz=FALSE for horizontal bars

dotchart(x) if x is a data frame, plots a Cleveland dot plot (stacked plots line-by-line and column-by-column)

pie(x) circular pie-chart

boxplot(x) "box-and-whiskers" plot

sunflowerplot(x, y) id. than plot() but the points with similar coordinates are drawn as flowers which petal number represents the number of points

stripplot(x) plot of the values of x on a line (an alternative to boxplot() for small sample sizes)

coplot(x | y | z) bivariate plot of x and y for each value or interval of values of z

interaction.plot (f1, f2, y) if f1 and f2 are factors, plots the means of y (on the y-axis) with respect to the values of f1 (on the x-axis) and of f2 (different curves); the option fun allows to choose the summary statistic of y (by default fun=mean)
The following parameters are common to many plotting functions:

- `termplot(mod.obj)`
- `stars(x)`
- `contour(x, y, z)`
- `qqnorm(x)`
- `plot.ts(x)`
- `pairs(x)`
- `mosaicplot(x)`
- `assocplot(x)`
- `fourfoldplot(x)`
- `matplot(x,y)`

Graphical parameters:

These can be set globally with `par(...)`: many can be passed as parameters to plotting commands.

- `adj` controls text justification (0: left-justified, 0.5 centred, 1: right-justified)
- `bg` specifies the colour of the background (ex.: `bg="red"`, `bg="blue"`, ...)
- `cex` controls the size of tick-labels and titles (and axis-labels)
- `cex.axis` specifies the size of texts and symbols with respect to the axes
- `cex.lab` specifies the size of the xy-labels and the title
- `cex.main` specifies the size of the sub-title
- `cex.sub` specifies the size of the sub-title
- `font` controls the style of text, `cex` specifies the size of text
- `lsep` controls the spacing between line-segments
- `lwd` controls the width of lines
- `lty` controls the type of lines, can be an integer or string (1: "solid", 2: "dashed", 3: "dotted", 4: "dotdash", 5: "longdash", 6: "twodash", or a string of up to eight characters (between "*" and "^*) which specifies alternatively the length, in points or pixels, of the drawn elements and the blanks, for example `lty="-4^4"` will have the same effect than `lty=2`
- `lwd` a numeric which controls the width of lines, default 1
- `mar` a vector of 4 numeric values which control the space between the axes
- `mfcol` a vector of the form `c(nr,nc)` which partitions the graphical window as a matrix of nr lines and nc columns, the plots are then drawn in columns
- `mtext` adds text given by `text` in the margin specified by `side` (see axis() below); `line` specifies the line from the plotting area
- `segments(x0, y0, x1, y1)` draws lines from points (x0,y0) to points (x1,y1)
- `arrows(x0, y0, x1, y1, angle= 30, code=2)` id. with arrows at points (x0,y0) if `code=2`, at points (x1,y1) if `code=1`, or both if `code=3`: angle controls the flank from the shaft to the edge of the arrow head
- `abline(a,b)` draws a line of slope b and intercept a
- `abline(h=y)` draws a horizontal line at ordinate y
- `abline(v=x)` draws a vertical line at abscissa x
- `polygon(x, y)` draws a polygon linking the points with coordinates given by x and y
- `legend(x, y, legend)` adds the legend at the point (x,y) with the symbols given by legend
- `title()` adds a title and optionally a sub-title
- `axis(side, vect)` adds an axis at the bottom (`side=1`), on the left (`side=2`), at the top (`side=3`), or on the right (`side=4`); `vect` (optional) gives the abcissa (or ordinate) where tick-marks are drawn
- `segments(x0, y0, x1, y1)` draws lines from points (x0,y0) to points (x1,y1)
- `rug(x)` draws the data on the x-axis as small vertical lines
- ` locator(n, type="n", ...)` returns the coordinates (x,y) after the user has clicked n times on the plot with the mouse; also draws symbols (type="p") or lines (type="l") with respect to optional graphical parameters (...); by default nothing is drawn (type="n")

Lattice (Trellis) graphics:

- `yplot(x)` bivariate plots with (many) functionalities
- `barchart(x)` histogram of the values of x with respect to those of y
- `dotplot(x)` Cleveland dot plot (stacked plots line-by-line and column-by-column)
- `densityplot(x)` density functions plot (smoothed histograms)
- `bwplot(x)` "box-and-whiskers" plot
- `qqmath(x)` quantiles of x with respect to the values expected under a theoretical distribution
- `stripplot(x)` single dimension plot
- `wireframe(x)` 3D wireframe plot
- `cloud(x)` 3D scatter plot
In the normal Lattice formula, \( y \sim g1 \times g2 \) has combinations of optional conditioning variables \( g1 \) and \( g2 \) plotted on separate panels. Lattice functions take many of the same arguments as base graphics plus also \texttt{data=} the data frame for the formula variables and \texttt{subset=} for subsetting. Use \texttt{panel=} to define a custom panel function (see \texttt{apropos(“panel”)} and \texttt{?panel}). Lattice functions return an object of class trellis and have to be \texttt{print}-ed to produce the graph. Use \texttt{print(xplot(...))} inside functions where automatic printing doesn’t work. Use \texttt{lattice.theme} and \texttt{lset} to change Lattice defaults.

**Optimization and model fitting**

\begin{align*}
\texttt{optim(par, fn, method = c(“Nelder-Mead”, “BFGS”, “CG”, “L-BFGS-B”, “SANN”) \quad \text{general-purpose optimization;}} \\
\texttt{par \quad \text{is initial values, } fn \quad \text{is function to optimize (normally minimize).}} \\
\texttt{nlm(f, p) \quad \text{minimize function } f \text{ using a Newton-type algorithm with starting values } p} \\
\texttt{lm(formula) \quad \text{fit linear models; formula is typically of the form response \texttt{~} termA + termB + ... \text{; use } I(x^2) \text{ for terms made of nonlinear components}} \\
\texttt{glm(formula, family=) \quad \text{fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution; family is a description of the error distribution and link function to be used in the model; see ?family}} \\
\texttt{nls(formula) \quad \text{nonlinear least-squares estimates of the nonlinear model parameters}} \\
\texttt{approx(x, y=) \quad \text{linearly interpolate given data points; } x \text{ can be an xy plotting structure}} \\
\texttt{spline(x, y=) \quad \text{cubic spline interpolation}} \\
\texttt{loess(formula) \quad \text{fit a polynomial surface using local fitting}} \\
\texttt{Many of the formula-based modeling functions have several common arguments: } \texttt{data=} \text{ the data frame for the formula variables, } \texttt{subset=} \text{ a subset of variables used in the fit, } \texttt{na.action=} \text{ action for missing values: } \texttt{”na.fail”}, \texttt{”na.omit”}, \text{or a function. The following generics often apply to model fitting functions:}} \\
\texttt{predict(fit,...)} \quad \text{predictions from fit based on input data} \\
\texttt{df.residual(fit)} \quad \text{returns the number of residual degrees of freedom} \\
\texttt{coef(fit)} \quad \text{returns the estimated coefficients (sometimes with their standard-errors)} \\
\texttt{residuals(fit)} \quad \text{returns the residuals} \\
\texttt{deviance(fit)} \quad \text{returns the deviance} \\
\texttt{fitted(fit)} \quad \text{returns the fitted values} \\
\texttt{logLik(fit)} \quad \text{computes the log likelihood of the model and the number of parameters} \\
\texttt{AIC(fit)} \quad \text{computes the Akaike information criterion or AIC}} \\
\texttt{Statistics} \\
\texttt{aov(formula)} \quad \text{analysis of variance model} \\
\texttt{anova(fit,...)} \quad \text{analysis of variance (or deviance) tables for one or more fitted model objects} \\
\texttt{density(x)} \quad \text{kernel density estimates of } x \\
\texttt{binom.test(). pairwise.t.test(). power.t.test(). prop.test().t.test().} \quad \text{use help.search(“test”)} \\
\texttt{Distributions} \\
\texttt{rnorm(n, mean=0, sd=1) \quad \text{Gaussian (normal)}} \\
\texttt{rexp(n, rate=1) \quad \text{exponential}} \\
\texttt{rgamma(n, shape, scale=1) \quad \text{gamma}} \\
\texttt{rpois(n, lambda) \quad \text{Poisson}} \\
\texttt{rweibull(n, shape, scale=1) \quad \text{Weibull}} \\
\texttt{rcauchy(n, location=0, scale=1) \quad \text{Cauchy}} \\
\texttt{rbeta(n, shape1, shape2) \quad \text{beta}} \\
\texttt{rt(n, df) \quad \text{‘Student’ (t)}} \\
\texttt{rf(n, df1, df2) \quad \text{Fisher–Snedecor (F)}} \\
\texttt{rchisq(n, df) \quad \text{Pearson}} \\
\texttt{rbinom(n, size, prob) \quad \text{binomial}} \\
\texttt{rgeom(n, prob) \quad \text{geometric}} \\
\texttt{rhyper(nn, m, n, k) \quad \text{hypergeometric}} \\
\texttt{rlogis(n, location=0, scale=1) \quad \text{logistic}} \\
\texttt{rnorm(n, meanlog=0, sdlog=1) \quad \text{lognormal}} \\
\texttt{rbinom(n, size, prob) \quad \text{negative binomial}} \\
\texttt{runif(n, min=0, max=1) \quad \text{uniform}} \\
\texttt{rweibull(nn, m, n), rsignrank(nn, n) \quad \text{Wilcoxon’s statistics}} \\
\texttt{All these functions can be used by replacing the letter } r \text{ with } d, p \text{ or } q \text{ to get, respectively, the probability density } \texttt{(dfunc(x, ...)), the cumulative probability density } \texttt{(qfunc(x, ...)), and the value of quantile } \texttt{(qfunc(p, ...), with } 0 \text{ < } p \text{ < 1).}} \\
\texttt{Programming} \\
\texttt{function( arglist ) expr \quad \text{function definition}} \\
\texttt{return(value)} \\
\texttt{if(cond) expr} \\
\texttt{if(cond) cons.expr else alt.expr} \\
\texttt{for(var in seq) expr} \\
\texttt{while(cond) expr} \\
\texttt{repeat expr} \\
\texttt{break} \\
\texttt{next} \\
\texttt{Use braces } \{} \texttt{around statements} \\
\texttt{ifelse(test, yes, no) \quad \text{a value with the same shape as } test \text{ filled with elements from either yes or no}} \\
\texttt{do.call(funnname, args) \quad \text{executes a function call from the name of the function and a list of arguments to be passed to it}}